Nourry Justine
Cherifi Sanâ

université PARIS-SACLAY

JAVA

Projet de Programmation :

Structure secondaire d'ARN

Comparaison de sous-arbres

Master 1 Bio-Informatique et Biostatistique (BIBS)
2020-2021

Table des matières

Introduction

L'informatique représente aujourd'hui un outil conséquent, voire indispensable dans la résolution de problématiques biologiques, et suscite ainsi un grand intérêt par son approche plus rapide et efficace sur de nombreuses thématiques.
Au cours de ce projet, nous nous sommes ainsi intéressées aux repliements en structures secondaires des molécules d'ARN.
Par définition, les Acide Ribonucléiques (ou ARN) sont des séquences génétiques simple brins formées de bases nucléiques : A, U, G, C.
Cette séquence va subir des repliements dans l'espace par la formation de liaisons entre couples de nucléotides (liaison deux à deux).
Ces repliements sont permis par l'action de diverses interactions énergétiques.
Les interactions entre acides aminés, régies par liaisons hydrogènes au sein d'une molécule d'ARN simple brin, vont en effet dicter la structuration de la future protéine et jouent un rôle dans sa fonction.

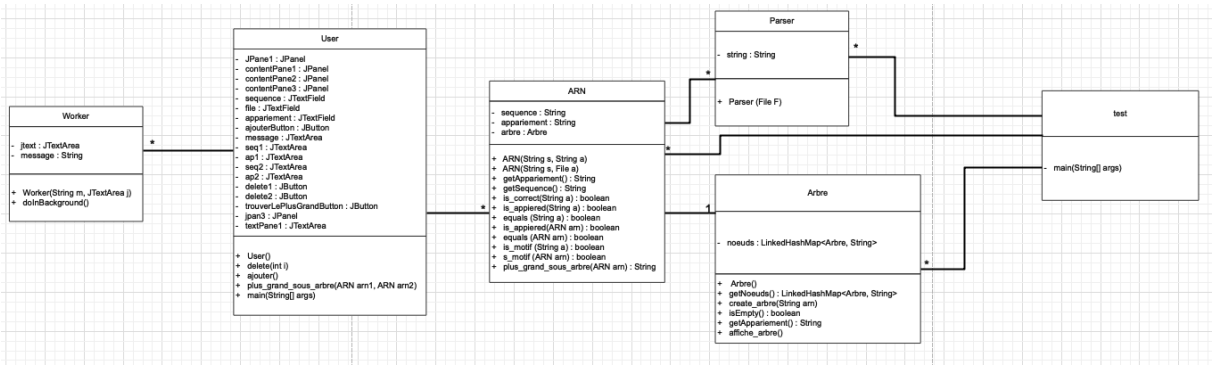
Trois différents niveaux de structure protéiques sont ainsi définis au sein de l'organisme.
Ces niveaux varient selon le type de protéine, le niveau de maturation d'une protéine, ou encore en fonction du milieu dans lequel la protéine se trouve.
On compte la structure primaire qui correspond à l'enchaînement de nucléotides, la structure secondaire définie par certaines liaisons formées entre ces nucléotides et la structure tertiaire représente la forme tridimensionnelle de la protéine

Ainsi, la structure secondaire des protéines se définit tel un réseau d'interactions locales entre résidus d'acides aminés, régies par la présence de liaisons hydrogènes.
Parmi cet ensemble d'appariements, on compte des régions en hélices (autrement dit les tiges) et des régions non appariées (les boucles).

Dans ce projet, nous avons cherché à représenter un brin d'ARN (acide ribonucléique), avec sa séquence et ses appariements (autrement dit les structures secondaires précédemment abordées) par un système de parenthésage, en utilisant un couple de parenthèses pour chaque couple de bases appariées et des tirets pour les bases non appariées (impliqués dans des boucles).
Nous allons nous focaliser sur la formation de ces structures secondaires pouvant être modélisées par des arbres par l'association de la séquence avec son appariement. Le but étant de comparer les séquences aux arbres
Le langage utilisé ici est le JAVA, un langage de programmation objet avec de nombreuses possibilités de création d'interfaces graphiques.
Le but final du projet étant de construire une interface graphique utilisateur dans le but de répondre efficacement à des besoins biologiques.

Diagramme des classes

Au cours du projet, nous nous sommes servis d'un diagramme des classes (UML).
Le but étant de mettre en avant les liens entre les différentes classes et objets intervenant dans notre code. (image disponible dans le git pour plus de clarté)



Descriptif des fonctionnalités développées

Nous avons créé un fichier Readme afin d'y répertorier les informations essentielles sur le projet et ainsi pouvoir suivre son avancée plus efficacement, les différents points relatifs à l'installation et la mise à jour du code, y étant relatés.

La première étape consistait en la création des différentes classes utiles pour le projet.
Nous avons ainsi créé la classe « ARN » avec ses constructeurs et ses get.
Il y a deux constructeurs, le premier prend en paramètres l'appariement et la séquence tandis que le deuxième prend en paramètres le fichier Stockholm afin de récupérer l'appariement et la séquence de l'ARN.
Au sein de cette classe, plusieurs méthodes ont ainsi été créées.
On citera ainsi la méthode *is_correct* qui vérifie si l'appariement est un arbre correct, la méthode *is_appiered/equals* afin de vérifier si l'ARN auquel on s'intéressait avait le même appariement que l'autre ARN.
Nous avons également créé la méthode *is-motif* testant si notre ARN (« this ») est contenu dans l'ARN.
Enfin, la méthode *plus_grand_sous_arbre*, récupère le plus grand sous-arbre commun entre this et ARN.

Une autre classe au rôle prépondérant est la classe « Arbre ». On y retrouve une méthode *create_arbre* qui comme son nom l'indique permet de créer un arbre à partir de l'appariement.
Nous avons également fait le test afin de savoir si l'arbre est vide grâce à is_empty.
La méthode get_appariement nous donne l'appariement à partir de l'arbre et enfin on a pu afficher les noeuds de l'arbre dans la console grâce à *affiche_arbre*

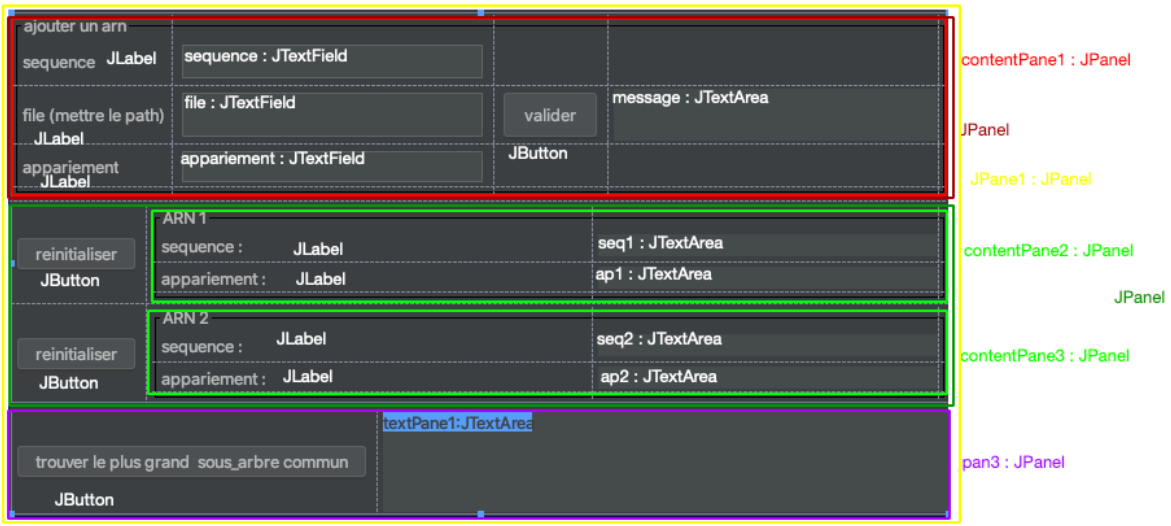
La classe Affichage, finalement remplacée par la classe User, a été également créée afin d'y recenser tous les éléments liés à l'interface graphique.

Nous avons créé une classe Parser afin de repérer la ligne contenant le motif souhaité et par la suite de remplacer chaque élément de la structure secondaire par son parenthésage associé.

Le Worker que nous avons créé est assez simple et permet d'afficher un message dans la JTextArea souhaité dans la fenêtre graphique.
Nous avons également choisi de coder une classe « test » afin de pouvoir faciliter le test de chaque méthode que nous utilisions, ce qui nous a permis de tester très régulièrement notre code.

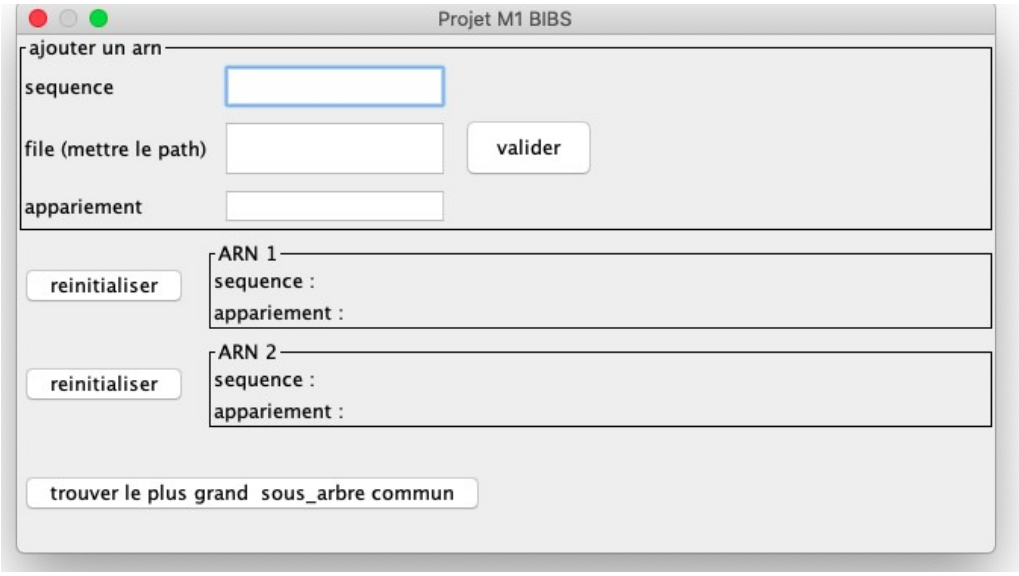
Maquette de l'interface utilisateur

L'interface utilisateur est composé d'une imbrication de JPanels ainsi que de JLabels, JButtons, JTextFields, JTextArea dans les différents JPanels.

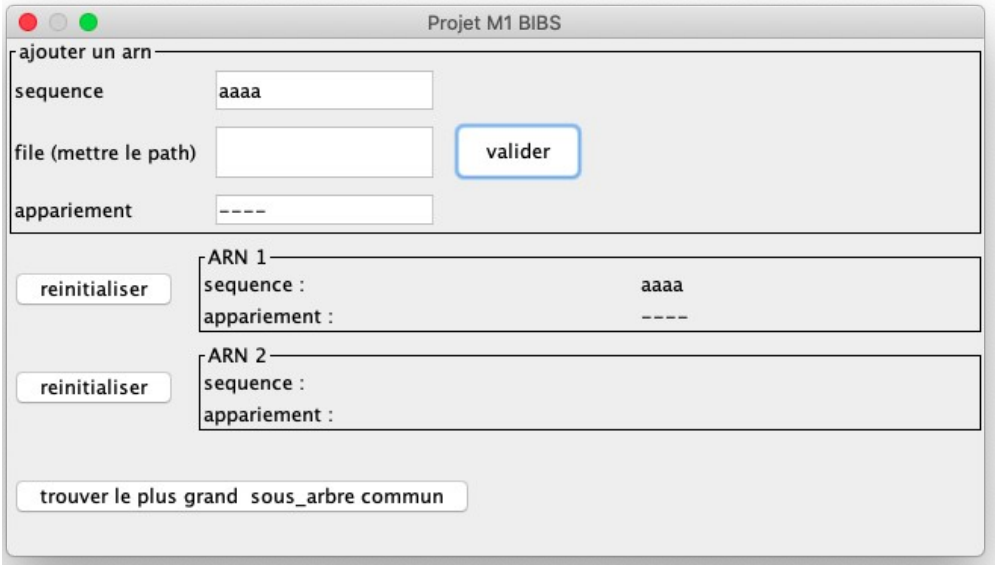


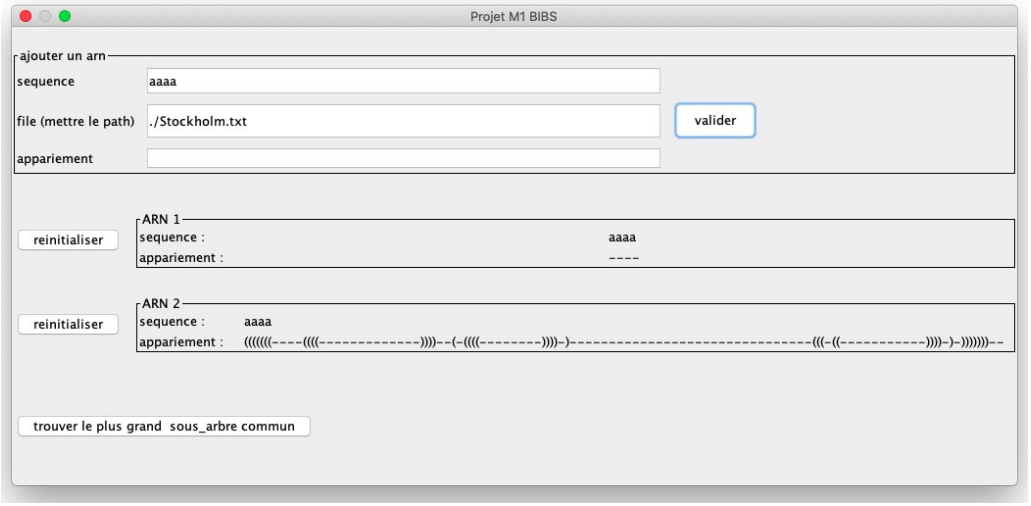
Maquette de l'interface utilisateur

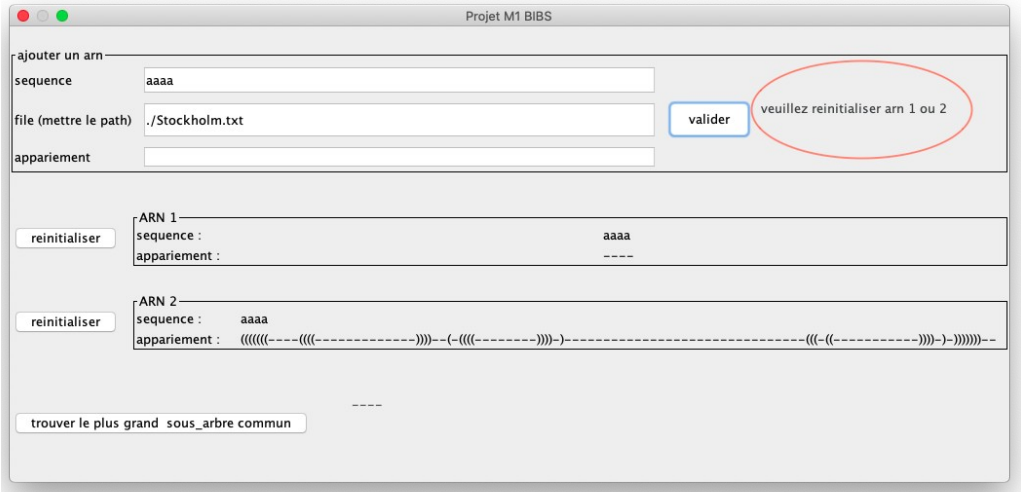
L'interface utilisateur est simple d'utilisation :



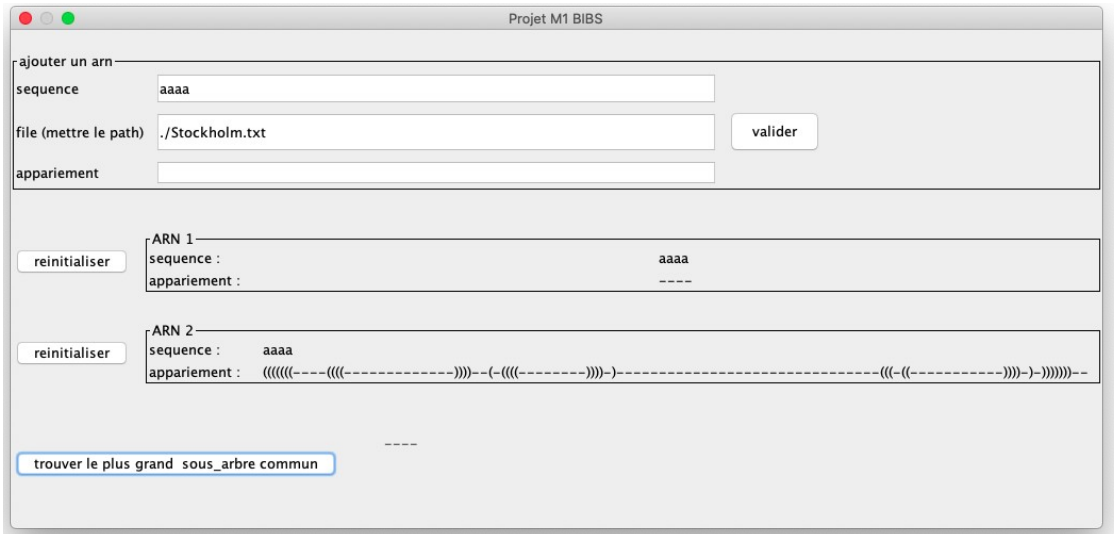
Dans la zone « ajouter un arn » il suffit de rentrer la séquence dans la zone prévue à cet effet ainsi que la séquence d'appariement ou le chemin du fichier au format stockholm puis de valider en appuyant sur le bouton « valider » pour ajouter un ARN. Après validation, soit un message d'erreur apparaît à côté du bouton « valider », soit la séquence et l'appariement s'affiche dans la première zone « ARN » de libre.







Lorsque l'on appuie sur le bouton « trouver le plus grand sous-arbre commun » une fois les 2 ARN sont définis, l'appariement du plus grand sous-arbre commun s'affichera alors à droite de ce bouton.



Nous avons des problèmes pour redimensionner la taille de la fenêtre graphique lorsque les séquences sont trop grandes. Plusieurs essais ont été fait mais aucun résultat probant donc il n'y a pas de code pour gérer la dimension de la fenêtre.

Environnement Technique (git, IDE)

Nous avons utilisé l'IDE IntelliJ pour écrire les fichiers java, pour les partager nous avons utilisé github (lien : https://github.com/justine66/projetM1).

Pour le diagramme de classe nous avons utilisé le logiciel EdrawMax.

Perspectives

Beaucoup de choses peuvent être améliorées. En effet la fenêtre graphique ne se redimensionne pas seule et on ne peut pas afficher les arbres, de plus il aurait pu être intéressant d'utiliser un parcoureur de fichier pour l'ajout de l'ARN à partir d'un fichier. Lorsqu'un utilisateur rentre une séquence et un appariement on ne test pas non plus si la séquence est de la même taille que l'appariement.

On considère également qu'il y a qu'un appariement possible pour le moment mais on pourrait réfléchir pour travailler avec une liste d'appariements.

Conclusion

Le projet que nous avons mené s'est avéré assez riches sur les possibilités et les options qui s'offraient à nous dans la création de cette interface.
Un premier défi a donc été d'apprendre à structurer et organiser nos idées sur la manière de procéder afin de créer une interface aussi efficace qu'agréable et simple d'utilisation pour les utilisateurs.

Nous nous sommes servis du logiciel de contrôle de version Git afin de soumettre très régulièrement l'avancée et du code.

Nous avons ainsi créé un répertoire de Travail, sur lequel nous déposions chaque nouvelle version accompagnée de commentaire sur les modifications venant d'être apportées, également nous précisions au fur et à mesure tous les problèmes rencontrés et les étapes auxquelles nous étions bloquées afin de mieux reprendre la fois suivante et ainsi faciliter le suivi des étapes de création du code.
Les commandes Git add, commit et push ont ainsi servi à mettre en ligne le travail et le rendre accessible.
Le projet n'a pas été simple à mener de bout en bout et nous avons dû faire face à plusieurs défis et difficultés dans sa conception, en particulier lorsque nous avons redimensionné la taille de fenêtre d'interface graphique. Également, lorsque nous voulions afficher le plus grand sous arbre, nous avons des problèmes à afficher un sous arbre correct.